



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии» (ИУ7)

Отчет по лабораторной работе №3 по курсу «Анализ алгоритмов»

«Трудоёмкость сортировок»

Группа: ИУ7-53Б

Студент:

(Подпись, дата)

Дьяченко А. А.
(Фамилия И. О.)

Преподаватель:

(Подпись, дата)

Строганов Д. В.
(Фамилия И. О.)

Москва, 2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Аналитическая часть	4
2 Конструкторская часть	5
2.1 Сортировка выбором	5
2.2 Сортировка перемешиванием	6
2.3 Модель вычислений	7
2.4 Трудоемкость алгоритмов	7
2.4.1 Сортировка выбором	7
2.4.2 Сортировка перемешиванием	8
3 Технологическая часть	9
3.1 Требования к ПО	9
3.2 Средства реализации	9
3.3 Сведения о модулях программы	9
3.4 Реализация алгоритмов	9
4 Исследовательская часть	11
4.1 Технические характеристики	11
4.2 Пример работы программы	11
4.3 Время выполнения реализованных алгоритмов	12
4.4 Занимаемая память реализованных алгоритмов	13
4.5 Вывод	13
ЗАКЛЮЧЕНИЕ	14
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	15

ВВЕДЕНИЕ

Сортировка — это последовательное расположение или разбиение на группы чего-либо в зависимости от выбранного критерия [1].

Любой алгоритм сортировки можно разбить на три основные части:

- сравнение элементов для определения их упорядоченности;
- перестановка элементов;
- сортирующий алгоритм, который осуществляет сравнение и перестановку элементов до тех пор, пока все элементы не будут упорядочены.

Цель лабораторной работы — изучить и исследовать трудоемкость алгоритмов сортировки.

Для достижения поставленных целей ставятся задачи:

- реализация требуемых алгоритмов сортировки: блонная, выбором, перемешиванием;
- сравнение требуемого времени выполнения реализуемых алгоритмов в тактах процессора и занимаемой памяти;
- подготовка отчёта по лабораторной работе.

1 Аналитическая часть

Блинная сортировка («pancakesort») — это не конкретный алгоритм сортировки, а класс сортировок, в которых допускается только одна операция — переворот элементов последовательности до какого-либо индекса [1].

При сортировке выбором из массива выбирается элемент с наименьшим значением и обменивается с первым элементом. Затем из оставшихся элементов снова выбирается элемент с наименьшим значением и обменивается со вторым элементом, и т.д. [2]. Сортировка выбором является частной реализацией блинной сортировки, поэтому в данном отчёте эти сортировки считаются тождественными.

По алгоритму сортировки перемешиванием указатель перемещается от левого края массива да правого и по необходимости меняет местами элементы в нужном порядке. После прохода слева направо указатель возвращается к началу массиву, так же по необходимости меняя местами элементы в соответствии с порядком [3].

2 Конструкторская часть

В этом разделе будут представлены схемы реализуемых алгоритмов.

2.1 Сортировка выбором

Схема алгоритма сортировки выбором представлена на рисунке 1.

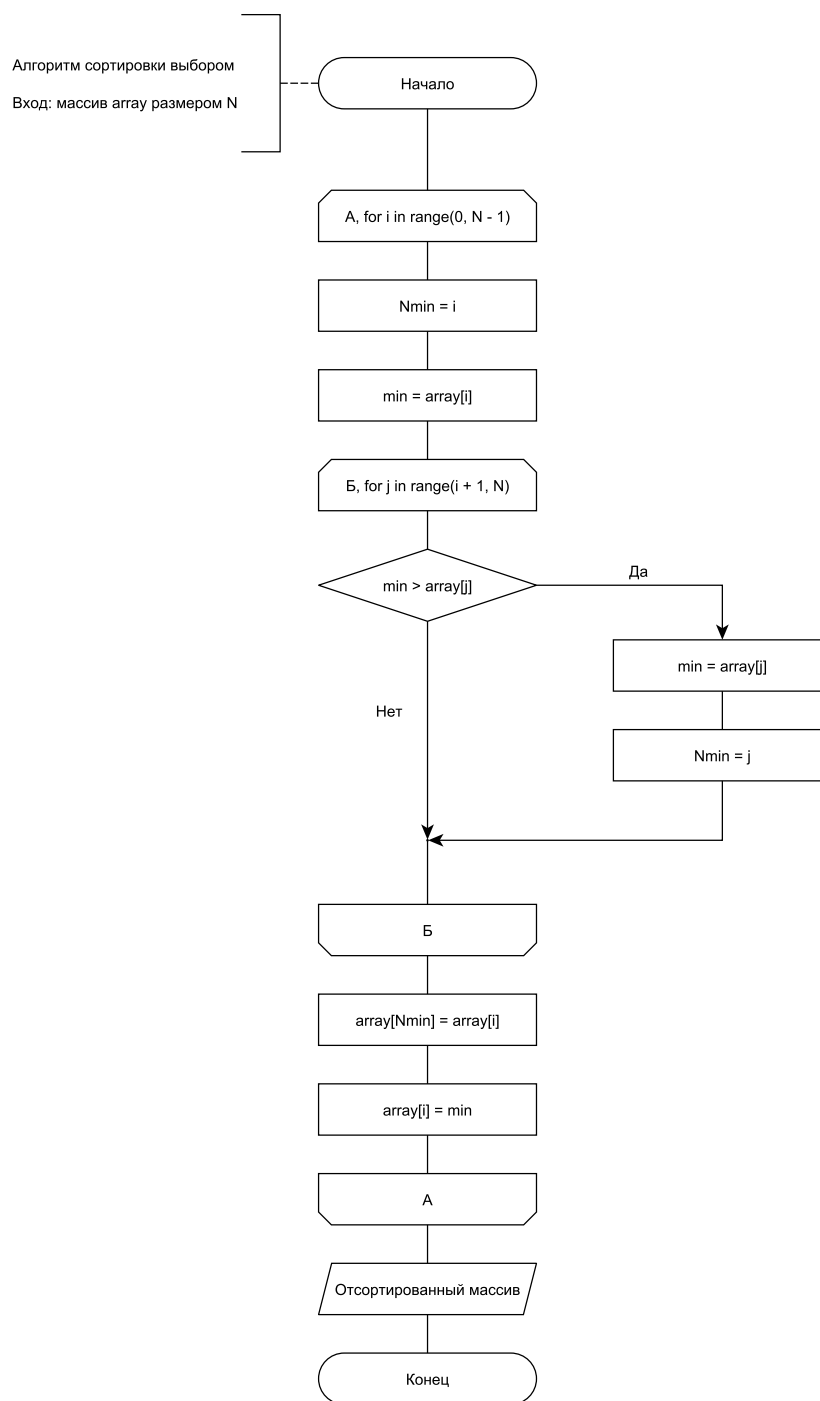


Рисунок 1 – Схема алгоритма сортировки выбором

2.2 Сортировка перемешиванием

Схема алгоритма сортировки перемешиванием представлена на рисунке 2.

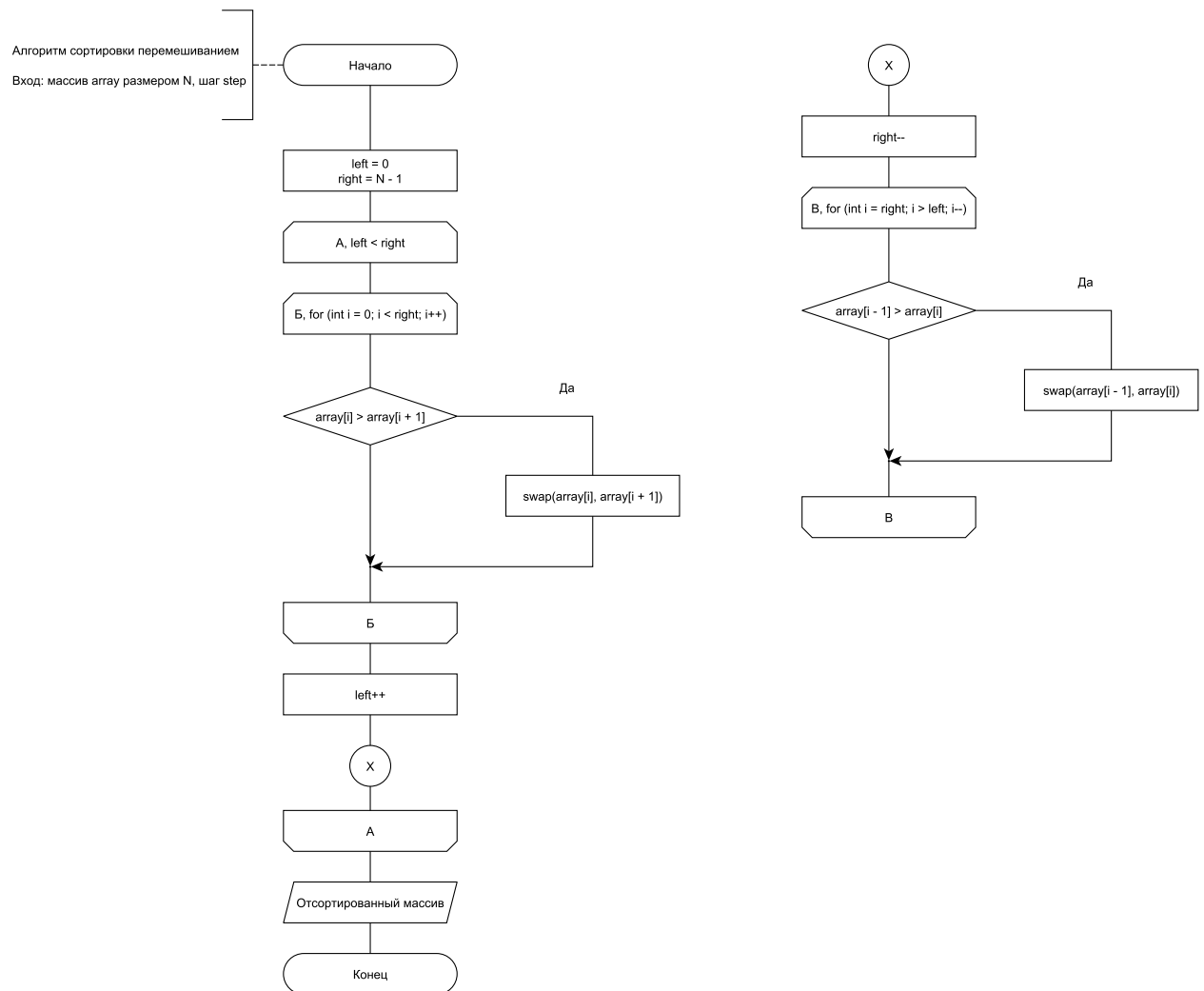


Рисунок 2 – Схема алгоритма сортировки перемешиванием

2.3 Модель вычислений

Для последующего вычисления трудоемкости необходимо ввести модель вычислений:

- а) операции из списка (1) имеют трудоемкость 1;

$$+, -, *, /, \%, ==, !=, <, >, <=, >=, [], ++, -- \quad (1)$$

- б) трудоемкость оператора выбора `if условие then A else B` рассчитывается как (2);

$$f_{if} = f_{условия} + \begin{cases} f_A, & \text{если условие выполняется,} \\ f_B, & \text{иначе.} \end{cases} \quad (2)$$

- в) трудоемкость цикла рассчитывается как (3);

$$f_{for} = f_{инициализации} + f_{сравнения} + N(f_{тела} + f_{инкремента} + f_{сравнения}) \quad (3)$$

- г) трудоемкость вызова функции равна 0.

2.4 Трудоемкость алгоритмов

В следующих частях будут рассчитаны трудоемкости представленных ранее алгоритмов сортировки.

Далее размер массива будет обозначаться символом N .

2.4.1 Сортировка выбором

Трудоёмкость сортировки выбором состоит из:

- трудоёмкости внешнего цикла по $i \in [0..N-1]$, трудоёмкость которого: $f = 2 + (N-1) \cdot (2 + f_{inner})$;
- трудоёмкости внутреннего цикла по $j \in [i+1..N]$, трудоёмкость которого в среднем: $f_{inner} = 2 + (N-1)/2 \cdot (2 + 2 + 2)$;

Итого, средняя трудоёмкость сортировки выбором:

$$f = 3N^2 - 1 \quad (4)$$

2.4.2 Сортировка перемешиванием

Трудоёмкость сортировки перемешивания состоит из:

- трудоёмкости внешнего цикла A , трудоёмкость которого: $f = 2 + (N - 1) \cdot (2 + f_{inner})$;
- трудоёмкости двух внутренних циклов по $i \in [0..N]$, трудоёмкость которых в среднем: $f_{inner} = 2 + (N - 1) \cdot (2 + 2 + f_{swap})$;
- трудоёмкости функции обмена $f_{swap} = 7$.

Итого, средняя трудоёмкость сортировки перемешиванием:

$$f = 11N^2 - 20N + 11 \quad (5)$$

Решение уравнения $11N^2 - 20N + 11 = 3N^2 - 1$ даёт ответ $N = 1.5$. Т.е. трудоёмкость сортировки перемешиванием больше трудоёмкости сортировки выбором после размера массива больше единицы.

3 Технологическая часть

В данном разделе приведены требования к программному обеспечению, средства реализации и листинги кода.

3.1 Требования к ПО

На вход подаётся массив. На выходе требуется получить отсортированный массив.

3.2 Средства реализации

В качестве языка программирования для реализации лабораторной работы был выбран C++. Данный выбор обусловлен наличием инструкции `rdtsc` [4] для замера количества тактов.

3.3 Сведения о модулях программы

Программа состоит из трех программных модулей:

- 1) `main.cpp` – главный модуль программы, содержащий меню;
- 2) `algo.cpp`, `algo.h` – модуль с реализацией алгоритмов;
- 3) `time.cpp`, `time.h` – модуль замера времени работы алгоритмов.

3.4 Реализация алгоритмов

В листинге 1 приведена реализация алгоритма сортировки выбором.

Листинг 1 – Сортировка выбором

```
1 void selection_sort(std::vector<int>& array) {
2     int n = array.size();
3
4     for (int i = 0; i < n - 1; ++i) {
5         int minIndex = i;
6         for (int j = i + 1; j < n; ++j) {
7             if (array[j] < array[minIndex]) {
8                 minIndex = j;
9             }
10        }
11        std::swap(array[i], array[minIndex]);
12    }
13 }
```

В листинге 2 приведена реализация алгоритма сортировки перемешиванием.

Листинг 2 – Сортировка перемешиванием

```
1 void shaker_sort(std::vector<int>& array) {
2     int n = array.size();
3     bool swapped;
4
5     do {
6         swapped = false;
7         for (int i = 0; i < n - 1; ++i) {
8             if (array[i] > array[i + 1]) {
9                 std::swap(array[i], array[i + 1]);
10                swapped = true;
11            }
12        }
13
14        if (!swapped) {
15            break;
16        }
17
18        swapped = false;
19        for (int i = n - 2; i >= 0; --i) {
20            if (array[i] > array[i + 1]) {
21                std::swap(array[i], array[i + 1]);
22                swapped = true;
23            }
24        }
25    } while (swapped);
26 }
```

4 Исследовательская часть

4.1 Технические характеристики

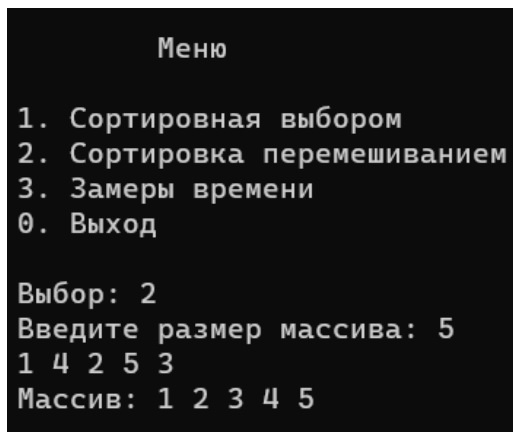
Технические характеристики устройства, на котором выполнялся замерный эксперимент:

- операционная система Windows 11;
- память 16 ГБ;
- процессор 3,6 ГГц 6-ядерный процессор AMD Ryzen 5000 series 5.

Замеры проводились на ноутбуке, включенном в сеть электропитания. Во время тестирования ноутбук был нагружен только интегрированной средой разработки и непосредственно выполняемой программой.

4.2 Пример работы программы

На рисунке 3 представлен пример работы программы.



```
Меню
1. Сортировка выбором
2. Сортировка перемешиванием
3. Замеры времени
0. Выход

Выбор: 2
Введите размер массива: 5
1 4 2 5 3
Массив: 1 2 3 4 5
```

Рисунок 3 – Пример работы программы

4.3 Время выполнения реализованных алгоритмов

Замеры времени работы реализованных алгоритмов для определенного размера массива проводились 1000 раз, при этом каждый раз значения массива генерировались случайно.

Для измерения тактового времени была использована инструкция rdtsc [4].

В качестве результата, представленного на графике 4, взято среднее время выполнения алгоритмов в тактах процессора для массивов размера от 1 до 100.

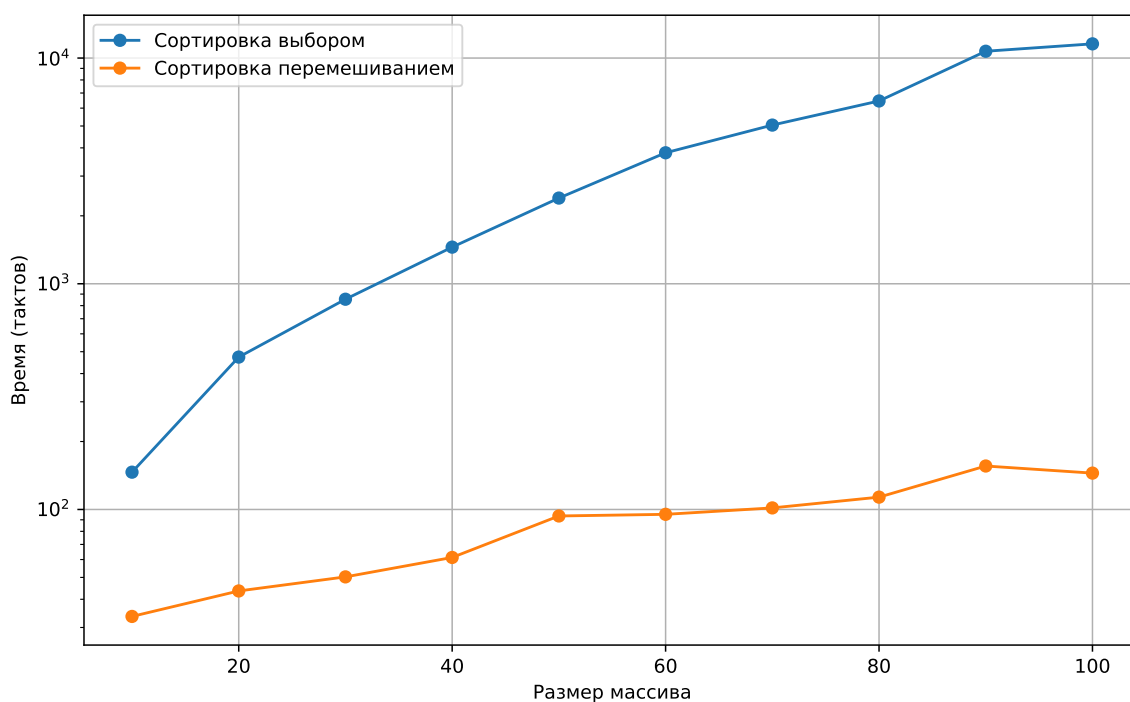


Рисунок 4 – Время выполнения алгоритмов

Сортировка выбором занимает значительно больше времени и уже на 10-ти элементах работает в 4.35 раза дольше, чем сортировка перемешиванием. Далее разница увеличивается экспоненциально.

4.4 Занимаемая память реализованных алгоритмов

Объём памяти, используемый алгоритмом сортировки целых чисел выбором, равен $4 * N$ байт. Алгоритм сортировки перемешиванием занимает на 1 байт больше памяти из-за бинарного флага *swapped*.

4.5 Вывод

В результате анализа замеров времени выполнения и затрат памяти на различных алгоритмах были сделаны следующие выводы:

- сортировка выбором работает медленнее, чем сортировка перемешиванием;
- отношение затраченного времени на сортировку массива увеличивается экспоненциально по мере увеличения размера массива;
- сортировка перемешиванием при любом размере массива занимает на 1 байт больше памяти, чем сортировка выбором.

ЗАКЛЮЧЕНИЕ

Цель работы достигнута: исследованы следующие алгоритмы сортировок:

- блинная;
- выбором;
- перемешиванием.

В ходе выполнения лабораторной работы были решены все задачи:

- реализованы требуемые алгоритмы;
- проведено сравнение требуемого времени выполнения реализуемых алгоритмов в тактах процессора и занимаемой памяти;
- подготовлен отчёт по лабораторной работе.

Не смотря на то, что рассчитанная трудоёмкость сортировки выбором меньше трудоёмкости сортировки перемешиванием при размере массива больше единицы, замеры показали, что сортировка выбором при любом размере сортируемого массива работает дольше, чем сортировка перемешиванием. Эта разница увеличивается экспоненциально по мере увеличения размера массива.

Занимаемая алгоритмами память отличается на один байт при любом размере массива.

Исходя из вышесказанного, в качестве алгоритма сортировки следует выбрать сортировку перемешиванием.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Вашинко А.* Блинная сортировка. — 2018.
2. *Сейдаметова З., Абдураманов З., Адильшаева Э.* Педагогическая таксономия алгоритмов сортировки вставками, выбором и разбиением // Ученые записки Крымского инженерно-педагогического университета. — 2015. — № 1. — С. 127—132.
3. *Прохоренко В. В., Погорелов Д. А., Кострицкий А. С.* СРАВНИТЕЛЬНЫЙ АНАЛИЗ ВРЕМЕНИ ВЫПОЛНЕНИЯ СОРТИРОВКИ ПО АЛГОРИТМУ ШЕЛЛА, ГНОМЬЕЙ СОРТИРОВКИ И СОРТИРОВКИ ПЕРЕМЕШИВАНИЕМ // Инновационные аспекты развития науки и техники. — 2021. — № 8. — С. 120—125.
4. *Microsoft.* RDTSC (Read Time-Stamp Counter). — URL: <https://learn.microsoft.com/ru-ru/cpp/intrinsics/rdtsc?view=msvc-170>.